CIRCUITOS LÓGICOS

Relatório VIII

Discente: *Rebeca de Macêdo Ferreira*. Matrícula: 2016000524

Nesse projeto fazemos a unidade VUA fatorada. Uma VUA faz o cálculo do "vai um antecipado" que nos auxiliará na construção do somador com Carry-Lookahead mais tarde. A implementação desta VUA foi feita seguindo o seguinte cálculo:

$$C_1 = G_0 + P_0 \cdot C_0,$$
$$C_2 = G_1 + P_1 \cdot C_1,$$
$$C_3 = G_2 + P_2 \cdot C_2,$$
$$C_4 = G_3 + P_3 \cdot C_3.$$

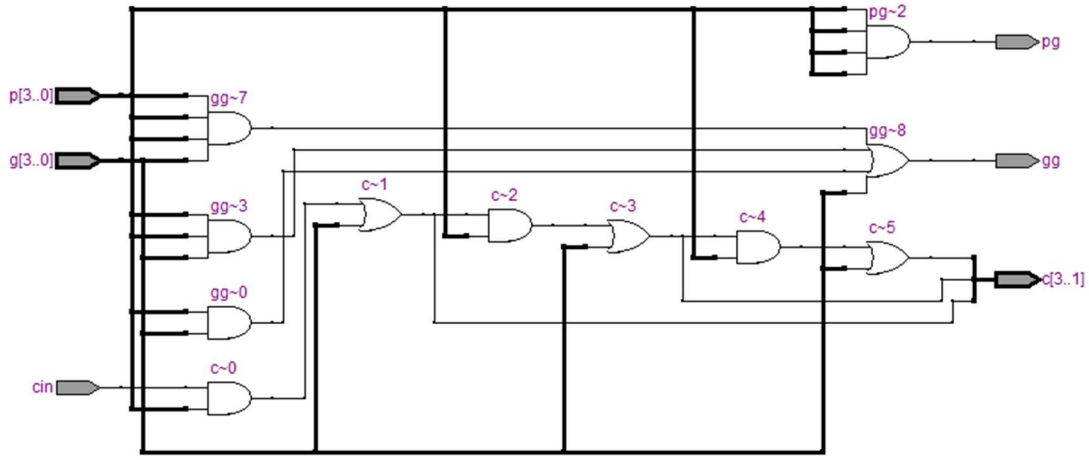
Além disso, é necessário fornecer a VUA os P's e G's para serem as saídas. Os P's e G's são calculados de acordo com as seguintes expressões:

$$PG = P_0 \cdot P_1 \cdot P_2 \cdot P_3,$$
$$GG = G_3 + G_2 \cdot P_3 + G_1 \cdot P_3 \cdot P_2 + G_0 \cdot P_3 \cdot P_2 \cdot P_1.$$

Assim, temos a implementação feita em VHDL:

```
1  library IEEE; use IEEE.STD_LOGIC_1164.ALL;
2
3  entity vuafatorada is
4      port(cin:    in STD_LOGIC;
5           p, g:   in STD_LOGIC_VECTOR(3 downto 0);
6           pg, gg: out STD_LOGIC;
7           c:      buffer STD_LOGIC_VECTOR(3 downto 1));
8  end;
9
10 architecture synth of vuafatorada is
11 begin
12     pg <= p(0) and p(1) and p(2) and p(3);
13     gg <= g(3) or (g(2) and p(3)) or (g(1) and p(3) and p(2)) or (g(0) and p(3) and p(2) and p(1));
14
15     c(1) <= g(0) or (p(0) and cin);
16     c(2) <= g(1) or (p(1) and c(1));
17     c(3) <= g(2) or (p(2) and c(2));
18 end;
```

Após compilar e corrigir pequenos erros de sintaxe da linguagem, obtivemos o seguinte diagrama de porta:



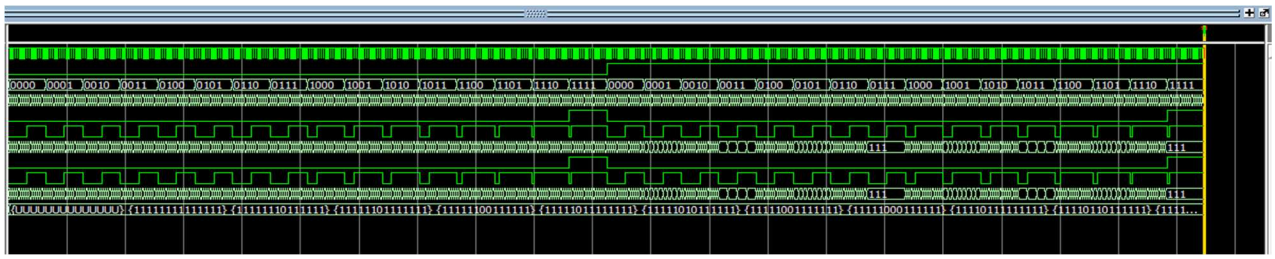
Para a simulação com o ModelSim, utilizamos o seguinte testvector:

```
0000_0000_0_0_0_000
0000_0001_0_0_0_001
0000_0010_0_0_0_010
0000_0011_0_0_0_011
0000_0100_0_0_0_100
0000_0101_0_0_0_101
0000_0110_0_0_0_110
0000_0111_0_0_0_111
0000_1000_0_1_0_000
0000_1001_0_1_0_001
0000_1010_0_1_0_010
0000_1011_0_1_0_011
0000_1100_0_1_0_100
0000_1101_0_1_0_101
0000_1110_0_1_0_110
0000_1111_0_1_0_111
0001_0000_0_0_0_000
0001_0001_0_0_0_001
0001_0010_0_0_0_010
0001_0011_0_0_0_011
0001_0100_0_0_0_100
0001_0101_0_0_0_101
0001_0110_0_0_0_110
```

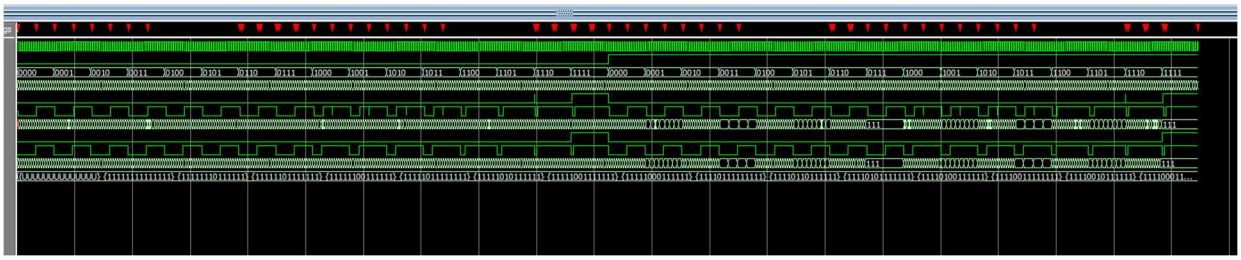
Testbench da VUA:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_arith.ALL;
4  use IEEE.STD_LOGIC_unsigned.ALL;
5  use STD.TEXTIO.ALL ;
6
7  entity testbench_vuafatorada is -- no inputs or outputs
8  end;
9  architecture sim of testbench_vuafatorada is
10     component vuafatorada
11         port (cin: in STD_LOGIC;
12               p, g: in STD_LOGIC_VECTOR(3 downto 0);
13               pg, gg: out STD_LOGIC;
14               c: buffer STD_LOGIC_VECTOR(3 downto 1));
15     end component;
16 signal clk: STD_LOGIC;
17 signal cin: STD_LOGIC;
18 signal p, g: STD_LOGIC_VECTOR(3 downto 0);
19 signal pg, gg: STD_LOGIC;
20 signal c: STD_LOGIC_VECTOR(3 downto 1);
21 signal pg_expected, gg_expected: STD_LOGIC;
22 signal c_expected: STD_LOGIC_VECTOR(3 downto 1);
23 constant MEMSIZE: integer := 512;
24 type tvarray is array (MEMSIZE downto 0) of
25 STD_LOGIC_VECTOR (13 downto 0);
26 signal testvectors: tvarray;
27 shared variable vectornum, errors: integer;
28 begin
29 -- instantiate device under test
30 dut: vuafatorada port map (cin, p, g, pg, gg, c);
31 -- generate clock
32 process begin
33     clk <= '1'; wait for 15 ns;
34     clk <= '0'; wait for 5 ns;
35 end process;
```

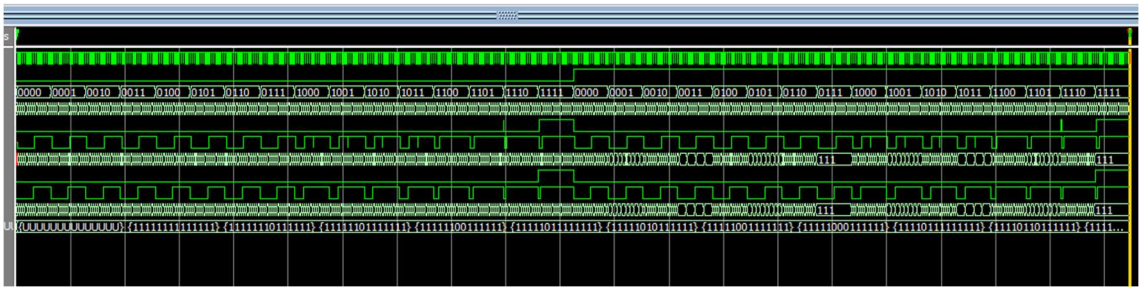
Simulamos o RTL level:



Simulamos o Gate level primeiro com o clk em 10 pra fazer os testes de tempo necessários e obtivemos o seguinte:



Nessa simulação ocorreram 120 erros. Então o clk foi aumentado para 15 ns, e obtivemos um resultado sem erros:



Com isso, portanto, temos uma VUA fatorada. Iremos agora construir uma VUA paralela, que deve ser mais rápida que a fatorada já que a fatorada faz o cálculo de cada C por vez.

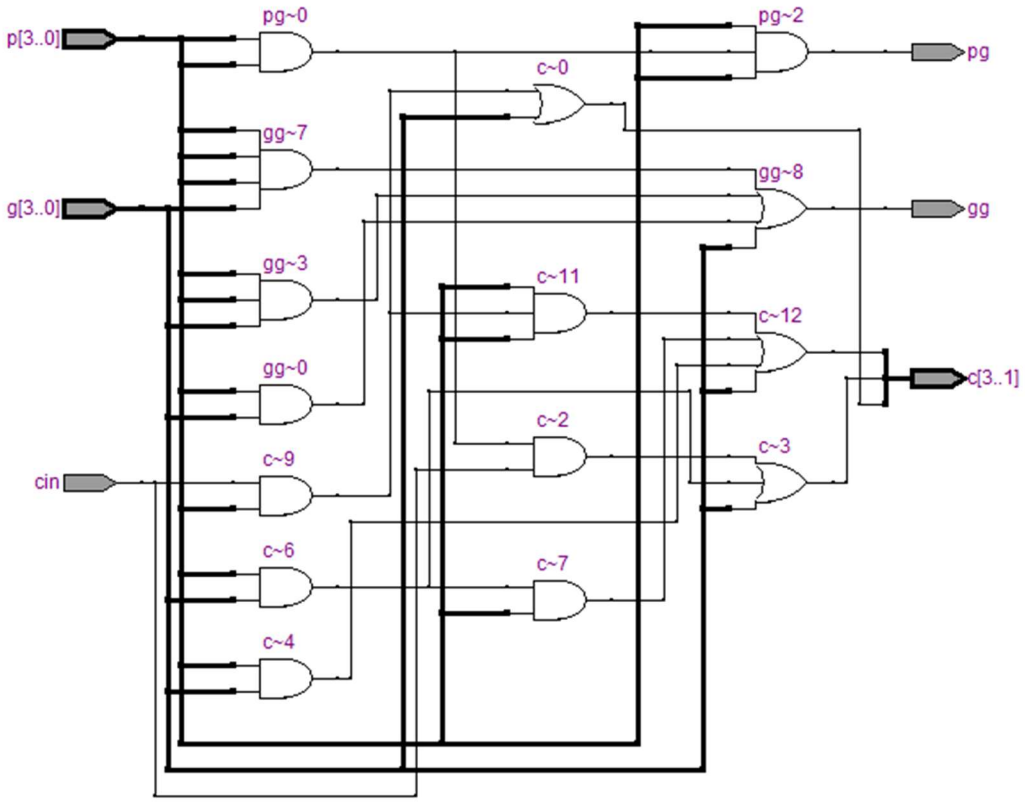
Para implementar o código em VHDL a seguinte expressão foi utilizada:

$$
\begin{aligned}
C_1 &= G_0 + P_0 \cdot C_0, \\
C_2 &= G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1, \\
C_3 &= G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + C_0 \cdot P_0 \cdot P_1 \cdot P_2, \\
C_4 &= G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3.
\end{aligned}
$$

Assim, construímos o VHDL da VUA Paralela

```
1     library IEEE; use IEEE.STD_LOGIC_1164.ALL;
2
3   entity vuaparalela is
4       port(cin:    in STD_LOGIC;
5            p, g:   in STD_LOGIC_VECTOR(3 downto 0);
6            pg, gg: out STD_LOGIC;
7            c:      buffer STD_LOGIC_VECTOR(3 downto 1));
8   end;
9
10  architecture synth of vuaparalela is
11  begin
12      pg <= p(0) and p(1) and p(2) and p(3);
13      gg <= g(3) or (g(2) and p(3)) or (g(1) and p(3) and p(2)) or (g(0) and p(3) and p(2) and p(1));
14
15      c(1) <= g(0) or (p(0) and cin);
16      c(2) <= g(1) or (g(0) and p(1)) or (p(0) and p(1) and cin);
17      c(3) <= g(2) or (g(1) and p(2)) or (g(0) and p(1) and p(2)) or (cin and p(0) and p(1) and p(2));
18  end;
```

Após compilado, obtivemos o RTL viewer

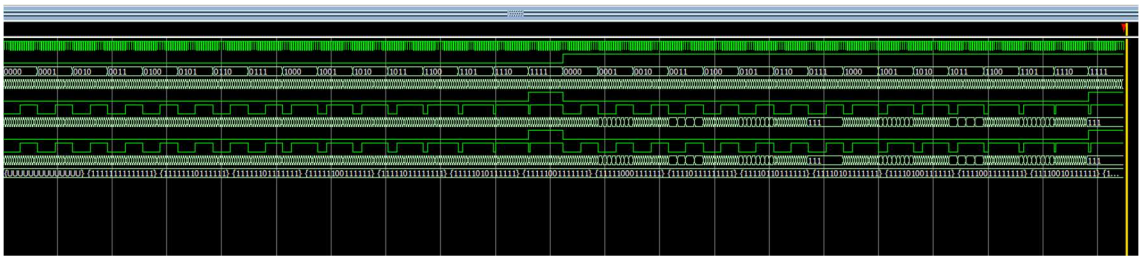


Para fazer as simulações com o ModelSim, foi utilizada o mesmo testvector do projeto da VUA fatorada.

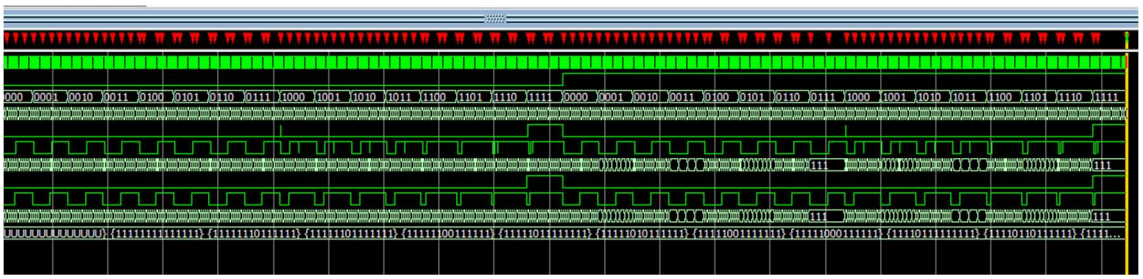
Testbench da VUA paralela:

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_arith.ALL;
4   use IEEE.STD_LOGIC_unsigned.ALL;
5   use STD.TEXTIO.ALL ;
6   entity testbench_vuaparalela is -- no inputs or outputs
7   end;
8   architecture sim of testbench_vuaparalela is
9       component vuaparalela
10          port (cin:    in STD_LOGIC;
11                p, g:   in STD_LOGIC_VECTOR(3 downto 0);
12                pg, gg: out STD_LOGIC;
13                c:      buffer STD_LOGIC_VECTOR(3 downto 1));
14      end component;
15  signal clk: STD_LOGIC;
16  signal cin: STD_LOGIC;
17  signal p, g: STD_LOGIC_VECTOR(3 downto 0);
18  signal pg, gg: STD_LOGIC;
19  signal c: STD_LOGIC_VECTOR(3 downto 1);
20  signal pg_expected, gg_expected: STD_LOGIC;
21  signal c_expected: STD_LOGIC_VECTOR(3 downto 1);
22
23  constant MEMSIZE: integer := 512;
24  type tvarray is array (MEMSIZE downto 0) of
25  STD_LOGIC_VECTOR (13 downto 0);
26  signal testvectors: tvarray;
27  shared variable vectornum, errors: integer;
28  begin
29  -- instantiate device under test
30  dut: vuaparalela port map (cin, p, g, pg, gg, c);
31  -- generate clock
32  process begin
33      clk <= '1'; wait for 15 ns;
34      clk <= '0'; wait for 5 ns;
35  end process;
```

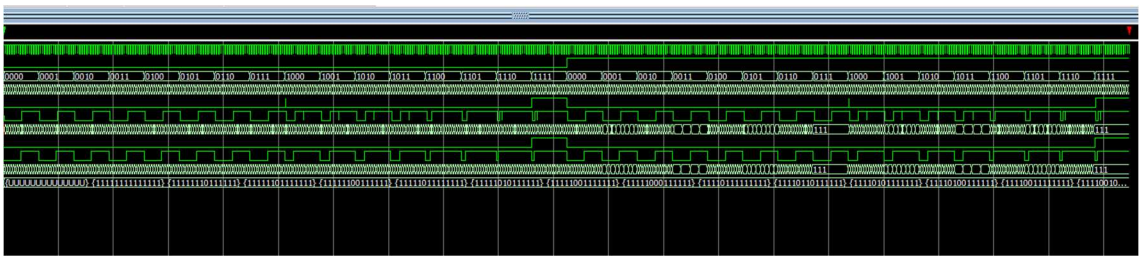
Fizemos a simulação do RTL level:



Na simulação do Gate level, diminuímos o valor pra 10ns para fazer os testes de tempo e obtivemos o seguinte:



Nessa simulação obtivemos 424 erros, então o clk foi aumentado pra 15 ns e obtivemos uma simulação sem erros.



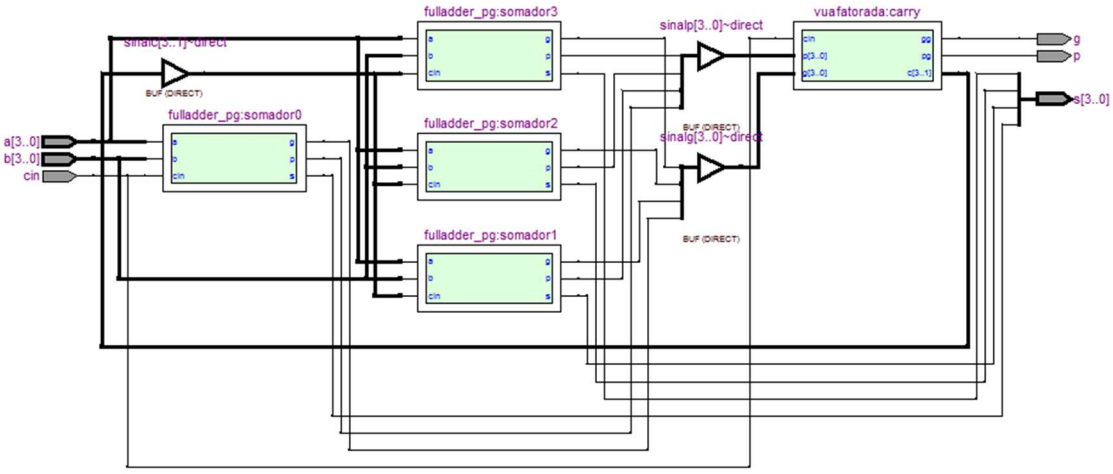
Como foi dito anteriormente, a VUA paralela é mais rápida que a VUA fatorada, nessas simulações o tempo de atraso do clk de ambas ficaram em 15 ns, isso pode está acontecendo por causa da versão do Quartus.

Somador com Carry lookahead

Primeiro vamos fazer um somador com a VUA fatorada, logo os arquivos VHDL da VUA fatorada e do somador completo com 'p' e 'g' devem estar neste projeto. Portanto o código em VHDL deste projeto ficou da seguinte maneira:

```
1  library IEEE; use IEEE.STD_LOGIC_1164.all;
2
3  entity cla4 is
4      port(a, b:         in  STD_LOGIC_VECTOR(3 downto 0);
5           cin:          in  STD_LOGIC;
6           s:            out STD_LOGIC_VECTOR(3 downto 0);
7           p, g:         buffer STD_LOGIC);
8  end;
9
10 architecture struct of cla4 is
11
12 component vuafatorada is
13     port(cin:    in STD_LOGIC;
14          p, g:   in STD_LOGIC_VECTOR(3 downto 0);
15          pg, gg: out STD_LOGIC;
16          c:      inout STD_LOGIC_VECTOR(3 downto 1));
17 end component;
18
19 component fulladder_pg is
20     port(a, b, cin:     in STD_LOGIC;
21          p, g: inout STD_LOGIC;
22          s: out STD_LOGIC);
23 end component;
24
25 signal sinalp, sinalg:   STD_LOGIC_VECTOR(3 downto 0);
26 signal sinalc:           STD_LOGIC_VECTOR(3 downto 1);
27
28 begin
29     somador0: fulladder_pg   port map (a(0), b(0), cin, sinalp(0), sinalg(0), s(0));
30     somador1: fulladder_pg   port map (a(1), b(1), sinalc(1), sinalp(1), sinalg(1), s(1));
31     somador2: fulladder_pg   port map (a(2), b(2), sinalc(2), sinalp(2), sinalg(2), s(2));
32     somador3: fulladder_pg   port map (a(3), b(3), sinalc(3), sinalp(3), sinalg(3), s(3));
33     carry:   vuafatorada port map (cin, sinalp, sinalg, p, g, sinalc);
34 end;
```

Após compilado obtivemos o diagrama de portas:

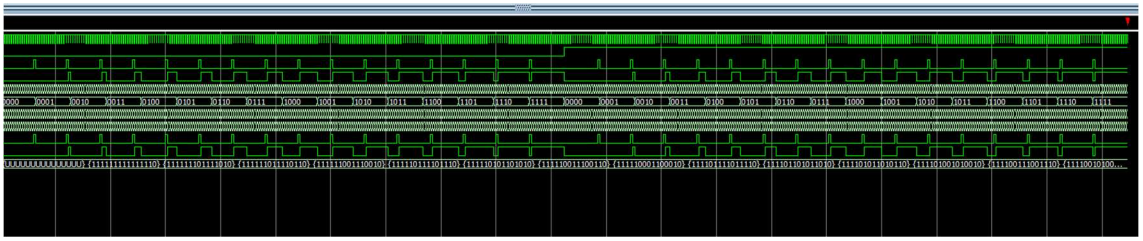


Para fazer as simulações, utilizamos o mesmo arquivo testvector utilizado anteriormente no projeto “vua fatorada”

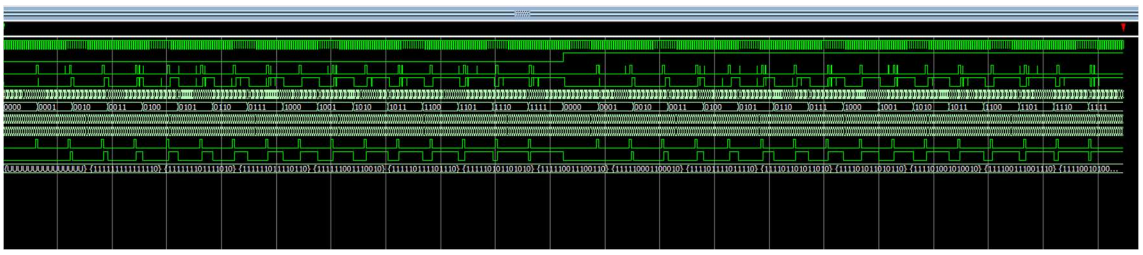
Testbench do Somador com Carry lookahead (VUA fatorada)

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
use STD.TEXTIO.ALL ;
entity testbench_cla4 is -- no inputs or outputs
end;
architecture sim of testbench_cla4 is
    component cla4
        port (a, b:  in  STD_LOGIC_VECTOR(3 downto 0);
              cin:  in  STD_LOGIC;
              s:   out STD_LOGIC_VECTOR(3 downto 0);
              p, g:  inout STD_LOGIC);
    end component;
signal clk: STD_LOGIC;
signal cin: STD_LOGIC;
signal p, g: STD_LOGIC;
signal s, a, b: STD_LOGIC_VECTOR(3 downto 0);
signal s_expected: STD_LOGIC_VECTOR(3 downto 0);
signal p_expected, g_expected: STD_LOGIC;
constant MEMSIZE: integer := 512;
type tvarray is array (MEMSIZE downto 0) of
STD_LOGIC_VECTOR (14 downto 0);
signal testvectors: tvarray;
shared variable vectornum, errors: integer;
begin
-- instantiate device under test
dut: cla4 port map (a, b, cin, s, p, g);
-- generate clock
process begin
    clk <= '1'; wait for 15 ns;
    clk <= '0'; wait for 5 ns;
end process;
```

Simulação RTL level:



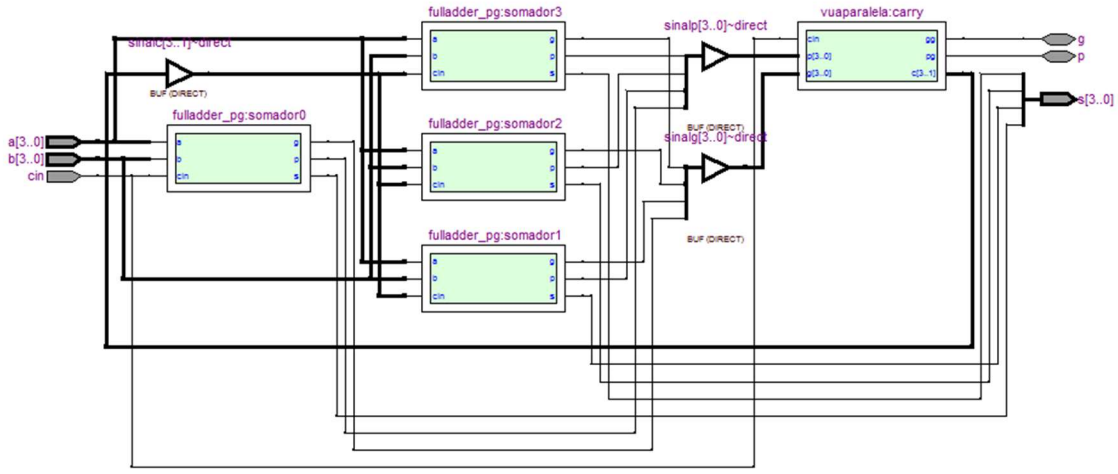
Na simulação Gate level os atrasos de clk não foram alterados, visto que já foi feito esses testes na contrução da VUA:



Agora iremos fazer as simulações do somador com a VUA paralela. Similar ao projeto anterior, fez-se necessário botar o arquivo VHDL da VUA paralela na pasta deste projeto, além do arquivo do somador completo com ‘p’ e ‘g’. O código VHDL deste projeto foi construído da seguinte maneira:

```
1   library IEEE; use IEEE.STD_LOGIC_1164.all;
2
3   entity cla4 is
4       port(a, b:          in  STD_LOGIC_VECTOR(3 downto 0);
5            cin:           in  STD_LOGIC;
6            s:             out STD_LOGIC_VECTOR(3 downto 0);
7            p, g:          inout STD_LOGIC);
8   end;
9
10  architecture struct of cla4 is
11
12  component vuaparalela is
13      port(p, g: in STD_LOGIC_VECTOR(3 downto 0);
14           cin: in STD_LOGIC;
15           c: inout STD_LOGIC_VECTOR(3 downto 1);
16           pg, gg: inout STD_LOGIC);
17  end component;
18
19  component fulladder_pg is
20      port(a, b, cin:     in STD_LOGIC;
21           p, g: inout STD_LOGIC;
22           s: out STD_LOGIC);
23  end component;
24
25  signal sinalp, sinalg:   STD_LOGIC_VECTOR(3 downto 0);
26  signal sinalc:           STD_LOGIC_VECTOR(3 downto 1);
27
28  begin
29      somador0: fulladder_pg   port map (a(0), b(0), cin, sinalp(0), sinalg(0), s(0));
30      somador1: fulladder_pg   port map (a(1), b(1), sinalc(1), sinalp(1), sinalg(1), s(1));
31      somador2: fulladder_pg   port map (a(2), b(2), sinalc(2), sinalp(2), sinalg(2), s(2));
32      somador3: fulladder_pg   port map (a(3), b(3), sinalc(3), sinalp(3), sinalg(3), s(3));
33      carry:   vuaparalela port map (sinalp, sinalg, cin, sinalc, p, g);
34  end;
```

Após compilado obtivemos o diagrama de portas:

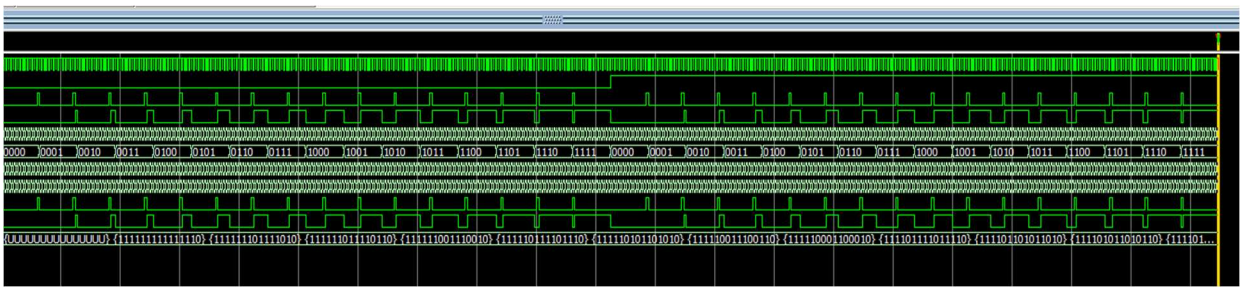


Para fazer as simulações, utilizamos o mesmo arquivo testvector utilizado anteriormente no projeto “vua fatorada”

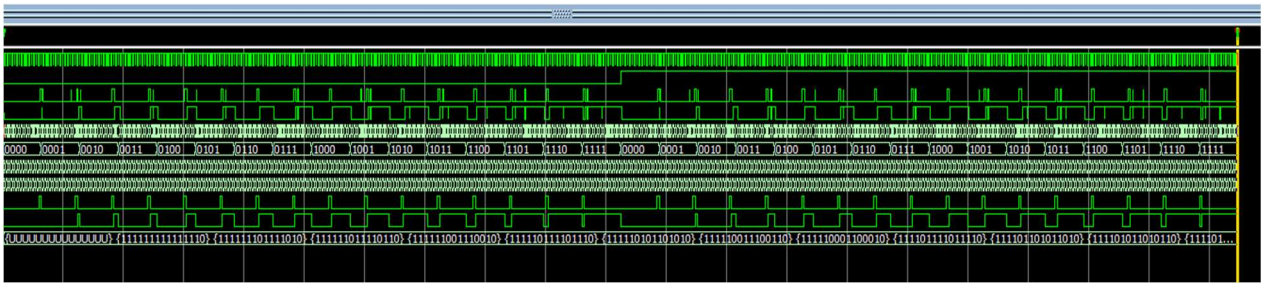
Testbench do Somador com Carry lookahead (VUA fatorada)

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_arith.ALL;
4   use IEEE.STD_LOGIC_unsigned.ALL;
5   use STD.TEXTIO.ALL ;
6   entity testbench_cla4 is -- no inputs or outputs
7   end;
8   architecture sim of testbench_cla4 is
9       component cla4
10          port (a, b:             in  STD_LOGIC_VECTOR(3 downto 0);
11                cin:          in  STD_LOGIC;
12                s:            out STD_LOGIC_VECTOR(3 downto 0);
13                p, g:         inout STD_LOGIC);
14      end component;
15  signal clk: STD_LOGIC;
16  signal cin: STD_LOGIC;
17  signal p, g: STD_LOGIC;
18  signal s, a, b: STD_LOGIC_VECTOR(3 downto 0);
19  signal s_expected: STD_LOGIC_VECTOR(3 downto 0);
20  signal p_expected, g_expected: STD_LOGIC;
21  constant MEMSIZE: integer := 512;
22  type tvarray is array (MEMSIZE downto 0) of
23  STD_LOGIC_VECTOR (14 downto 0);
24  signal testvectors: tvarray;
25  shared variable vectornum, errors: integer;
26  begin
27  -- instantiate device under test
28  dut: cla4 port map (a, b, cin, s, p, g);
29  -- generate clock
30  process begin
31      clk <= '1'; wait for 15 ns;
32      clk <= '0'; wait for 5 ns;
33  end process;
34  -- at start of test, load vectors
```

Simulação RTL level:



Simulação Gate level:



Ditetórios:

VUA fatorada

VHDL: VUA Fatorada/vuafatorada.vhd

Testbench: VUA Fatorada/testbench/testbench_vuafatorada.vhd

Test vector: VUA Fatorada/simulation/ModelSim/vuafatorada

VUA Paralela

VHDL: VUA Paralela/vuaparalela.vhd

Testbench: VUA Paralela/testbench/testbench_vuaparalela.vhd

Test vector: VUA Paralela/simulation/ModelSim/vuaparalela

CLA com VUA fatorada

VHDL(s): CLA_Fatorada/vuafatorada.vhd – fulladder_pg.vhd – cla4.vhd

Testbench: CLA_Fatorada /testbench/testbench_cla4.vhd

Test vector: CLA_Fatorada /simulation/ModelSim/cla4

CLA com VUA Paralela

VHDL(s): CLA_Paralela/vuaparalela.vhd – fulladder_pg.vhd – cla4.vhd

Testbench: CLA_Paralela /testbench/testbench_cla4.vhd

Test vector: CLA_Paralela /simulation/ModelSim/cla4